MENTAL POKER

Luka Seničić
ls7043@student.uni-lj.si
Faculty of Computer and Information Science, University of Ljubljana

Abstract. Poker is one of the most popular gambling games in nowadays and today it is most commonly played on the internet. Mental poker is a name for playing trusted poker game online with help of various cryptographic tools. This paper specifies requirements for online poker protocol, describes typical cryptographic problems when constructing a new one and gives examples of two protocols that were firstly introduced in order to play poker online, one requires Trusted Third Party (TTP) and one is TTP-free.

1 INTRODUCTION

The poker game was developed during an early 19th century and has been popular since then. From that time a lot has changed, we have made a huge progress since and a great amount of new things were invented. Two inventions that stand out in last half of century are computers and internet. When computers arrived people started to transfer various stuff in digital form. It turned out to be a useful thing because it required a less physical space and it was easily transferable so putting stuff on the computer became normal, same happened with games. Playing poker locally against computer is very easy to implement, but if we want to do it online against another person, things start to get more interesting. Poker is a game where a great deal of outcome is determined by luck and luck is produced by random events that are part of the game. In general, random events are extremely easy to forge without being physically present to witness them because they happened without any pattern so proving them over a long distance is hard. In that situation questions like “Are the cards shuffled fairly?”, “Did you draw cards without changing?”, “Do you have cards that you claim to have?” become vitally important when trying to implement poker over the internet. There comes this paper which presents answers to those questions by presenting protocols that are used in the online poker game. First, we will give a brief introduction to poker rules and formally look at the problem when trying to implement mental poker. In section three we look at some cryptographic background of mental poker and later we check two basic examples of protocols and analyze them.

2 POKER

2.1 Rules

The game of poker has many different rulesets for playing; probably most famous version is Texas Hold’Em. Most of the mental poker protocols are written in terms of five-card draw poker game (although all protocols can easily adapt for any type of game) because it is considered the simplest, so here we will give short introduction and rules for it.

Five-card draw variant that is commonly played among new players and rarely in casinos and tournaments. This game is specific because it is only one where player gets all cards dealt at the beginning and then improves them later by replacing. Here we will simplify the rules and remove parts that apply paying the blinds (small amounts of money payed in advance to play each the round).

At the beginning dealer shuffles the deck and deals five cards, all face down, to each player starting from the first left of the dealer. Then, when everyone gets all cards, the betting round starts. In betting round every player can bet on his cards and everybody can either raise an amount of money previous player has a bet or who don't want to pay simply fold and do not participate again until the next round. When all the bets are made and if more than one player still remains playing after the betting round than the "draw" round starts. Each player specifies how much cards he wants to replace and discards them. Now again each player is dealt the number of cards he has discarded (so again everybody would have five cards). When everybody gets their cards, another betting round is done. At the end, everybody still playing show their cards and the strongest hand takes all the money from the betting. One round of the game is now finished. Table showing how much is every hand strong can be found at [8].

2.2 Mental Poker

Mental poker is a name for poker game played without any physical elements (e.g. on-line). Each mental poker protocol must assure that game is being played fairly and as we have seen in the introduction, that is not an easy task. Formal requirements and properties which every new protocol should satisfy were written in 1985. by Claude Crépeau[9]:

- Uniqueness of cards: Before start of the game, players should be able to verify if the cards are right (that there are no duplicates); this is the easiest rule to achieve.
- Uniform random distribution of cards: We want shuffling cards to be done random and without any influences from players.
- Cheating detection with a high probability: Protocol must be able to if some player is attempting to cheat (seeing a face-down card)
- Complete confidentiality of cards: No player can't obtain any partial or complete information about cards that are faced down.
- Minimal effect of coalitions: Our protocol needs to minimize effects of coalitions between players, e.g. if one player tells another what cards does he have
- Complete confidentiality of strategy: If the player folds or loses at the end he has the option not to show cards what he owned. That is what we want in mental poker protocol as well, we will see later that this is probably the hardest requirement to satisfy.

Crépeau also wrote one additional requirement and that is Absence of trusted third party. This condition is completely real because it is not the best solution to rely on TTP, since every human can be bribed, but as we will see later it is easier to ignore this requirement because protocols with TTP are simpler and more efficient. However, we want TTP to participate in the game as little as possible.

3 CRYPTOGRAPHIC TOOLS

In this section, we present some difficultly solvable cryptographic problems which later will be used in designing new protocols. The list of them should be longer but we will present only this three, two of them are used Shamir-Rivest-Adelman protocol that will be mentioned later and the other, zero-knowledge proof is vitally important when designing any mental poker protocol.

3.1 Factorization Problem

Factorization problem is one of the more common ones in cryptography. The definition is very simple: for the given integer we want to find any integer (not even needs to be prime) that divides it. Fastest algorithm today for solving it is "general number field sieve". But here we will focus on "Dixon's random squares algorithm".≤

Dixon's random squares algorithm was published in 1981 by John Dixon. As many factoring algorithms it is based on finding two numbers x and y, such that $x \neq \pm y \pmod{n}$, but $x^2 \equiv y^2 \pmod{n}$. From that, we see that n divides $(x-y)$ or $(x+y)$. This is the hardest part of the algorithm and because it would take too much time generate numbers that are squares from integer, we will instead generate numbers that satisfy the weaker condition; their squares can be factored into small primes (find numbers whose squares are B-smooth).

For the beginning let's take some set $B = \{p_1,\dots,p_b\}$ called factor base which consists of b smallest prime numbers. Next we want to find at least b numbers (in practice a little more than b, let's say $b+4$) which squared factorization consists only from elements in set B. Finding those numbers can be done at random, but instead it is more useful to try numbers of the form $j + \lceil \sqrt{kn} \rceil$, $j \in \mathbb{N}_0$, $k \in \mathbb{N}$ and $\lfloor \sqrt{kn} \rfloor$, $k \in \mathbb{N}$. Let's suppose we obtained $c=b+4$ such numbers and we can write them in c congruences:

$$z_j^2 \equiv p_1^{a1j} \cdot p_2^{a2j} \cdot \dots \cdot p_b^{abj} \pmod{n},$$

where $1 \leq j \leq c$. For each j we want to write vector:

$$a_j = (a_{1j} \bmod 2, \dots a_{bj} \bmod 2) \in (\mathbb{Z}_2)^b,$$

so we want to find linear depended subset (over $\mathbb{Z}_2$) of these vectors a_j, that dependence must exist because we have c (more than b) vectors $j \in \{1,\dots,b\}$ and can be found with Gaussian elimination. The corresponding z_j will use each factor in B even number of times so we found our wanted numbers x and y from the beginning of the story. The time complexity of Dixon's random squares algorithm is:

$$O(e^{(1+O(1))\sqrt{\ln n * \ln(\ln n)}})$$

As we have said before factorization problem is one of the more common ones appearing in mental poker protocols and cryptography in general. If we could find a solution for it a many today popular cryptosystems, like RSA which is used often in mental poker as well will be in danger.

3.2 Discrete Logarithm Problem

Discrete logarithm can be defined as follows: Let's assume that $(G, \cdot)$ is multiplicative group of order n. Given $\beta \in \langle\alpha\rangle$, find a unique exponent a, $0 \leq a \leq n-1$, such that $\alpha^a \equiv \beta$. As factorization, discrete logarithm problem is one of the bases for many cryptosystems today. Although they are distinct problems and currently no efficient algorithm is known for solving them (on non-quantum computers), algorithms we do have for one problem are often adapted to the other. These similarities can be noticed between previously described "Dixon's random square algorithm" for factorization problem and "Index calculus method" we will now describe for discrete logarithm problem.

As before, in Index calculus method we also have factor base $B = \{p_1,\dots,p_b\}$. The first step here is precomputation stage where we need to compute logarithms of b primes in B. For that again we want to construct at least b congruences (in practice let's take $c = b + 10$ congruences) which have the following form:

$$\alpha^{xj} \equiv p_1^{\alpha 1j} \cdot p_2^{\alpha 2j} \cdot \dots \cdot p_b^{\alpha bj} \pmod{p}$$

where $1 \leq j \leq c$. This congruence is equivalent to:

$$x_j \equiv \alpha_{1j} \log p_1 \cdot \alpha_{2j} \log p_2 \cdot \dots \cdot \alpha_{bj} \log p_b \pmod{p-1}.$$

We have c congruences of this form and now we can get to values of $\log p_1, \dots, \log p_b$ by solving that system of c equations with Gaussian elimination. We are done with precomputation stage and logarithms of primes in factor base are computed. To solve actual problem $\log_\alpha \beta$ we start choosing random integers s until we can factor $\beta\alpha^s$ only with primes that are in the factor base, i.e. until we get this:

$$\beta\alpha^s \equiv p_1^{c1} \cdot p_2^{c2} \cdot \ldots \cdot p_b^{cb} \pmod p$$

and this congruence is equivalent to:

$$\log_\alpha\beta + s \equiv c_1 \log_\alpha p_1 \cdot c_2 \log_\alpha p_2 \cdot \ldots \cdot c_b \log_\alpha p_b \pmod{p-1}.$$

Since all coefficients are known except of $\log_\alpha\beta$, we can easily get it. Running time for precomputation stage is same as running time of Dixon's random squares:

$$O(e^{(1+O(1))\sqrt{\ln p * \ln(\ln p)}})$$

and the running time of the actual algorithm is:

$$O(e^{(1/2+O(1))\sqrt{\ln p * \ln(\ln p)}}).$$

Like factorization, discrete logarithm problem is also vitally important when designing cryptosystems and mental poker protocols. It is suitable for us because there is no known algorithm for computing it, yet inverse operation, exponentiation, can be done efficiently. That fact makes it well useful for zero-knowledge proofs (next part). In example later we will see how much first TTP-free protocol depends on discrete logarithm problem.

3.3 Zero-knowledge proof

The zero-knowledge proof is a technique which allows one person (prover) prove some statement to another (verifier) without revealing any partial information about statement except the fact that it is indeed true. If proving the statement requires some knowledge about the statement, the verifier can't prove the same statement to anyone else because he doesn't know the secret information about the statement. Now we will give the properties that zero-knowledge must satisfy and later we will give two examples for complete verification of discrete logarithm. The properties that every proof must satisfy are:

- Completeness: if the statement is true, the honest verifier will be convinced
- Soundness: if the statement is false, probability of accepting the lie must be negligible
- Polynomial time: the verifier must do private computation in polynomial time

Proof of knowledge of a discrete logarithm

Let p be a prime number where $p = 2q + 1$ where q is also a prime number. Following protocol will convince verifier that given $\beta = g^\alpha \bmod p$, the prover knows α.

- The prover sends $\beta = g^\omega \bmod p$ to verifier for some random $\omega \in \mathbb{Z}_q$
- Verifier responds by sending random challenge $c \in \mathbb{Z}_q$
- The prover responds with $r = \omega + \alpha c \bmod p$
- Verifier checks whether $g^r \bmod p == a\beta^c \bmod p$

Proof of equality of discrete logarithm

Let again p be a prime number where $p = 2q + 1$ and q is also prime. Given $u = g^\alpha \bmod p$ and $v = y^\beta \bmod p$. Now we want to convince the verifier that the prover knows α, β and $\alpha = \beta$ holds, where g and y have order q.

- The prover sends $(a, b) = (g^\omega, g^\omega)$ to the verifier for some random value $\omega \in \mathbb{Z}_q$
- Verifier responds by sending random challenge $c \in \mathbb{Z}_q$
- The prover responds with $r = \omega + \alpha c \bmod p$
- Verifier checks wether $g^r \bmod p == au^c \bmod p$ and $y^r \bmod p == bv^c \bmod p$

4 PROTOCOL EXAMPLES

Now, we will look at the two actual examples of poker protocols. All protocols are divided into ones that use TTP and one that are TTP-free. Today there are plenty of examples for both categories and it is impossible to choose one that is universal (most web-sites do not even publish data about it). That is why I decided to present and analyze two protocols which are first of its kind. We will analyze protocols based on a number of messages that are being sent, their length and whether or not they satisfy formal requirements written in Section 2.

4.1 Fortune-Merrit Protocol

The first mental poker protocol that uses TTP was published in 1984. by Stephen Fortune and Michael Merrit and it is called Fortune-Merrit protocol [5]. This was a huge improvement over everything that was invented until then because it permitted any number of players and uses inexpensive but it had one major difference, a trusted third party that in this case is called Card Salesman. Card Salesman is much like the manufacturer of a deck of cards, he only participates in the protocol at the preparation stage for the actual play so its computational cost is minimal.

This protocol requires the use of one-way functions (hashes) and permutations. A one-way function (OWF later) is a function that is easy to compute for every element of a domain and hard to invert when given an image of random input. As usual, the Card Salesman will use OWF to authenticate information he computes. This protocol assumes the availability of two network services: sending secret messages between pairs of players and broadcasting message to all players. Here we give a description for 3 players (Alice, Bob, and Charles) but the schema can easily be generalized to an arbitrary number of players.

Card Shuffling:

- Card Salesman randomly chooses permutation π
- Each player chooses three permutations (here he chooses three because that is the number of players that participates, if four players played then each would choose four); Alice chooses α_A, α_B, α_C, Bob chooses β_A, β_B, β_C and Charles chooses γ_A, γ_B, γ_C; all chosen permutations are then transmitted to Card Salesman and their encryptions using OWF are broadcasted
- Card Salesman computes and broadcasts the products $\delta_A=\beta_A^{-1}\gamma_A^{-1}\alpha_A^{-1}\pi^{-1}$, $\delta_B=\gamma_B^{-1}\alpha_B^{-1}\beta_B^{-1}\pi^{-1}$ and $\delta_C=\alpha_C^{-1}\beta_C^{-1}\gamma_C^{-1}\pi^{-1}$ (and their OWF values like before)

Now the Card Salesman job is finished and the game is ready to start. Notice that function δ_A that will Alice use, δ_B that will Bob use and δ_C are all public knowledge, only private parts are function chosen by each player (e.g. α_A, α_B, α_C chosen by Alice) but computations of OWF are known so they can be checked later. Now we describe Drawing cards part for Charles, all other are exactly the same, just we need to change the order of computations.

Drawing Cards (for Charles):

- Charles randomly chooses $y = \pi(x)$ that is not in any players hand and broadcasts y and $\delta_C(y)$
- firstly Alice computes and broadcasts $\alpha_C(\delta_C(y))$ (so she can negate inverse α_C^{-1})
- Bob computes and broadcasts $\beta_C(\alpha_C(\delta_C(y)))$
- Charles computes $x = \gamma_C(\beta_C(\alpha_C(\delta_C(y))))$
- all players record that Charles has $y = \pi(x)$

We need to notice that order in which we make computations are very important here and it changes depending on what player is a drawing card. All actions to the end of the game can be made using this two

"subprotocols". At the end of the game to prove correctness of game every player needs to reveal their private permutations then everyone can make all computations on their own to check whether the game was played fairly (since computed values of OWF of private permutations are known from the beginning we are assured that private functions can't be changed during the game).

Protocol analysis

First, we need to notice that only one player and Card Salesman needs to be fair in order to game to be correct. From formal requirements above we see that only Complete confidentiality strategy is not satisfied because all cards are reviled at the end no matter what.

When shuffling cards Card Salesman needs to broadcast n messages in total. That message consists of every δ_i (where i is every player) that he computed and its corresponding value of OWF. Each player needs to send a Card Salesman basicaly only one message that consists of all his private functions (e.g. for Alice that would be α_i for all players i) and their respective OWF values. That OWF values he also needs to broadcast.

Next, in drawing card stage, we will look separately player who draws card and everybody other. When player draws card he broadcasts one message consisting of value y and $\delta_i(y)$. Then every player computes his private part and broadcasts forward computations that he has made. That gives n messages in total (also notice that no OWF values are sent anymore).

Finally, we can conclude that with this protocol we can certainly play a fair game of poker (as long as Card Salesman and one player are fair) although still a lot of improvements can be made. Probably most noticeable thing is that here no trail is left and players can't challenge somebody at any moment if they suspect that he cheated, they all have to wait until the end of the round.

4.2 Shamir-Rivest-Adelman Protocol

In 1981. Shamir, Rivest and Adelman (inventors of RSA-encryption) published [3] first ever protocol for playing poker over a long distance (later called SRA protocol). Their paper is very interesting because first, they give strictly mathematical proof that such thing can't be achieved and later they present a new protocol (using RSA encryption) that disputes their proof. This protocol is for two players only. It is based on commutative RSA cryptosystem (we will see that in practice we can use any commutative cryptosystem) and the discrete logarithm problem.

Let us assume that two players P_1 and P_2 want to play poker over a telephone. They first need to agree on public large prime p. Then P_1 chooses own private key k_1 (so it has an inverse in Z_{p-1}) and P_2 chooses private key k_2 (also needs to have an inverse in Z_{p-1}). Cards are represented in values $D=\{x_1,...,x_{52}\}$ where every x_i is $1 < x_i < p$ (they both know values in set D). Now we finished initialization stage and we can move on to describing protocols.

Card Shuffling (P_2 is a dealer):

- P_2 needs to compute $C = \{ c_1,..,c_{52} \}$ where $c_i = x_i^{k2} \bmod p$ for every x_i
- Next P_2 randomly chooses permutation $\pi \in S_{52}$ (which he doesn't send to P_1) and permutes set C to get set $C^* = \{ c_{\pi(1)},.., c_{\pi(52)} \}$
- P_2 sends set C^* to P_1

Now the P_1 has permuted and encrypted cards. The Drawing cards part is little different form before because P_1 will draw card for himself and for P_2.

Drawing Cards (all computations are made in modulo p):

- P_1 first chooses card for opponent; he randomly chooses i $\in$ {1,...,52} and sends encrypted card $c_i \in C^*$ to P_2 who can now decrypt card with his own private key
- Next P_1 will randomly chooses j $\in$ {1,...,52}
- P_1 encrypts card $c_j \in C^*$ with his own private key to get $E_{k2}(cj) = (c_j)^{k1} = x_j^{k2*k1} = c_j'$ and sends it to P_2
- P_2 decrypts c_j' with his private key and gets $D_{k2}(c_j') = x_j^{k2*k1*k2\wedge(-1)} = x_j^{k1*k2*k2\wedge(-1)} = x_j^{k1}$ and sends that information to P_1
- P_1 now decrypts that information with his private key and gets $D_{k1}(x_j^{k1}) = x_j^{k1*k1\wedge(-1)} = x_j$

Since this description is just for drawing one card, same procedure can be repeated any number of times. At the end P_1 and P_2 both reveal their own private keys to prove that they didn't cheat.

Protocol analysis

This protocol has some obvious flaws. First only two players can play with this protocol. From formal requirements in Section 2 there are three that are not filled: Cheating detection with high probability, Complete confidentiality of cards and Complete confidentiality of strategy. Argument for the last one is same as in previous example. Complete confidentialty of cards isn't holding up also because if they is not careful and marks some x_i that are Quadratic residues and other that are not then person drawing cards will have some knowledge about cards he draws (because $x_i \in$ *mod p* QR iff x_i^a *mod p* $\in$ QR). Here, cheating depends on the careless implementation of the SRA protocol, e.g. we need to be very careful when choosing prime p.

Number of messages sent here is very small. In the shuffling part of the protocol P_2 does all the work and at the end, he only sends the final result to P_1. So we have only one message that consists from 52 encrypted cards. Drawing cards is also very cheap. P_1 needs to send two messages in total, one he chooses for the opponent and one he chooses for himself (size of the message is obviously very small), while P_2 needs to send only one message that is also small, card for opponent on which he has decrypted his part.

We can conclude that SRA protocol is very cheap in terms of sending messages and their sizes, but it comes with the price. Security is the main problem of this protocol. Although I haven't found any TTP-free protocol which satisfies all formal requirements written above, there are some (like Crépeu's protocol) that are more computationally demanding but also more secure.

5 CONCLUSION

We introduced mental poker problem, explained why it is hard and very interesting from the cryptographic point of view. That is why we focused on two cryptographic problems we learned in class (factorization & discrete logarithm) and introduced zero-knowledge proof because it is very important when planning mental poker protocol. In protocol examples shown we see how much they depend on these cryptographic problems. A similar situation is with other TTP-free protocols which also rely on them. That could be a problem when first quantum computers arrive since we have quantum algorithms to break them. Today there are more than 600 web-sites available to play on-line poker and PokerStars (the biggest today) has around 70 million registered users. These numbers tell us that we have come a long way in making secure mental poker protocols since the beginning and for now we can safely play online poker over the internet.

REFERENCES

[1] M. Trampuš „Miselni poker: diplomska naloga“, Univerza v Ljubljani, 2008.

[2] H. Stamer „Bibliography on Mental Poker ver. 1.6“, 2007.

[3] A. Shamir, R. Rivest & L. Adelman „Mental Poker“, 'The Mathematical Gardner', pag. 37-43, 1981.

[4] J. Castella-Roca „Contributions to Mental Poker“, PhD thesis, Universitat Autonoma de Barcelona, 2005.

[5] S. Fortune & M. Merritt „Poker Protocols“, Advances in Cryptology – CRYPTO '84, pp. 454-464, 1984.

[6] I. Barany & Z. Füredi „Mental Poker with Three or More Players“, Information and Control, 59(1-3):84-93, 1983.

[7] D. Stinson „Cryptography: Theory and Practice Third edition“, CRC Press, 2006.

[8] C. Crépeau, “A Secure Poker Protocol that Minimizes the Effect of Player Coalitions”, Advances in Cryptography-CRYPTO ’85, pag. 73-86, 1985.

[9] Poker hand strength: http://www.wsop.com/poker-hands/